IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	Docket No.: **TIF-32157**
Gilbert Cabillic	Art Unit: **2122**
Serial No.: **10/003,570**	Examiner: **Steelman, Mary J.**
Filed: **10/24/2001**	Conf. No.: **5072**
For: **DATA PROCESSING APPARATUS, SYSTEM AND METHOD**	

APPELLANTS' BRIEF – 37 C.F.R. § 1.192(c)

Commissioner for Patents

Alexandria, VA 22313-1450

Dear Sir:

This Appeal Brief is submitted in connection with the above-identified application in response to the final Office Action mailed December 28, 2005.

I.   REAL PARTY IN INTEREST

Texas Instruments Incorporated is the real party in interest.

II. RELATED APPEALS AND INTERFERENCES

Appellants are unaware of any related appeals and interferences.

III. STATUS OF CLAIMS

Final rejection of Claims 1-15 and 18-24 was made by the Examiner in the Office Action dated December 28, 2005. Claims 1-15 and 18-24 are on appeal. Claims 1-15 and 18-24 are reproduced in the Appendix to Appellants' Brief filed herewith.

IV. STATUS OF AMENDMENTS

Applicants submitted to the USPTO an Amendment 37 CFR § 1.116 on May 17, 2006. In an Advisory Action mailed on January 17, 2007, Examiner states that for purposes of appeal the amendment will be entered.

Applicants submit herewith a second Amendment 37 CFR § 1.116 in which they request that the term "desired" be removed from Claims 1 and 3 for clarity. Accordingly, the status of this amendment is pending.

V.   <u>SUMMARY OF THE CLAIMED SUBJECT MATTER</u>

A system and method for generating program code for translating high level code into instructions for one of a plurality of target processors comprises first determining a program code characteristic corresponding to a target processor. Then, selecting one or more predefined program code modules from a plurality of available program code modules in accordance with said desired program code characteristic, and generating program code for translating high level code into instructions for said target processor from said selected one or more predefined program code modules. Preferably, the method comprises forming agglomerated program code from a plurality of program code modules in accordance with said program code characteristic.

More specifically, Claim 1 requires and positively recites, a method for generating program source code for translating high level code into instructions for a target processor, the method comprising:

determining a program code characteristic corresponding to said target processor;

deriving one or more program code modules in accordance with said ~~desired~~ program code characteristic; and

generating program source code for translating high level code into instructions for said target processor from said one or more program code modules.

Claim 9 requires and positively recites, a method for creating program source code for translating between high level code and instructions for a target processor, comprising the steps of:

determining a program code characteristic corresponding to said target processor;

selecting one or more predefined program code modules in accordance with said program code characteristic; and

forming program source code for translating high level code into instructions for said target processor from said selected one or more predefined program code modules.

Claim 11 requires and positively recites, a data processing apparatus for creating program source code for translating between high level code and instructions for a target processor, the data processing apparatus being configured to:

determine a program code characteristic corresponding to said target processor identifier input to said data processing apparatus;

derive one or more program code modules in accordance with said program code characteristic; and

create program source code for translating high level code into instructions for said target processor from said derived one or more program code modules.

Claim 13 requires and positively recites, an apparatus, comprising at least one program source code module of a plurality of program source code modules for translating between high level code and instructions for a target processor, said at least one program code module corresponding to a characteristic of said target processor and being selected from said plurality of program source code modules.

Claim 22 requires and positively recites, a processor, configured in accordance with program code comprising at least one program source code module of a plurality of program source code modules, for translating between high level code and instructions for a target processor, said at least one program source code module being in accordance with a characteristic of said target processor and selected from said plurality of program source code modules.

Claim 23 requires and positively recites, a processor, configured by program code comprising an agglomeration of two or more program source code modules of said plurality of said program source code modules.

Claim 24 requires and positively recites, a system comprising a first and second processor, said first and second processor configured in accordance with program code

comprising at least two program source code modules, wherein the first of said at least two program source code modules is arranged to translate high level code to instructions for said first processor and a second of said at least two program source code modules is arranged to translate high level code to instructions for said second processor.

VI.   GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Are Claims 1-15 and 18-24 patentable under 35 U.S.C. 103(a) as being unpatentable over US Patent 6,571,388 B1 to Venkatraman et al., in view of US Patent 6,230,307 B1 to Davis et al.?

VII.   ARGUMENTS

Claims 1-15 and 18-24 stand rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent 6,571,388 B1 to Venkatraman et al., in view of US Patent 6,230,307 B1 to Davis et al. Applicants respectfully traverse this rejection as set forth below.

Independent Claim 1, as amended, requires and positively recites, a method for **generating program source code** for **translating high level code into instructions for a target processor**, the method comprising: "determining a program code characteristic corresponding to said target processor", "deriving one or more program code modules **in accordance with said program code characteristic**" and "**generating program source code for translating high level code into instructions for said target processor from said one or more program code modules**".

Independent Claim 9, as amended, requires and positively recites, a method for **creating program source code** for **translating between high level code and**

instructions for a target processor, comprising the steps of: "determining a program code characteristic corresponding to said target processor", "selecting one or more predefined program code modules **in accordance with said program code characteristic**" and "**forming program source code for translating high level code into instructions for said target processor from said selected one or more predefined program code modules**".

Independent Claim 11, as amended, requires and positively recites, a data processing apparatus for **creating program source code** for **translating between high level code and instructions for a target processor**, the data processing apparatus being configured to: "determine a program code characteristic corresponding to said target processor identifier input to said data processing apparatus", "derive one or more program code modules **in accordance with said program code characteristic**" and "**create program source code for translating high level code into instructions for said target processor from said derived one or more program code modules**".

Independent Claim 13, as amended, requires and positively recites, an apparatus, comprising **at least one program source code module** of a plurality of **program source code modules** for **translating between high level code and instructions for a target processor**, said at least one program source code module corresponding to a characteristic of said target processor and being selected from said plurality of program code modules.

Independent Claim 22, as amended, requires and positively recites, a processor, configured in accordance with **program code** comprising at least one **program source code module** of a plurality of **program source code modules**, for **translating between high level code and instructions for a target processor**, said at least one **program source code module** being in accordance with a characteristic of said target processor and selected from said plurality of **program source code modules**.

Independent Claim 23, as amended, requires and positively recites, a processor, configured by **program code** comprising an **agglomeration of two or more program source code modules** of said plurality of said program code modules.

Independent Claim 24, as amended, requires and positively recites, a system comprising a first and second processor, said first and second processor configured in accordance with program code comprising **at least two program source code modules**, **wherein the first of said at least two program source code modules is arranged to translate high level code to instructions for said first processor** and **a second of said at least two program source code modules is arranged to translate high level code to instructions for said second processor**.

In contrast, Venkatraman's pre-load analyzer determines which classes are needed -- NOT a program code characteristic. When Venaktraman does select a source of classes, it is according to available resources in the target devices -- NOT in accordance with desired code characteristics. As such, Venkatraman fails to teach or suggest, "deriving one or more program code modules **in accordance with said program code characteristic**", as required by Claim 1, OR ; and "said step of deriving comprises deriving respective program code modules **in accordance with said respective program code characteristics**", as required by Claim 5, OR ; "selecting one or more predefined program code modules **in accordance with said program code characteristic**", as required by Claim 9, OR "derive one or more program code modules **in accordance with said program code characteristic**", as required by Claim 11, OR "said at least one program code module **corresponding to a characteristic of said target processor** and being selected from said plurality of program source code modules", as required by Claim 13, OR "... said at least one program source code module **being in accordance with a characteristic of said target processor** and selected from said plurality of program source code modules", as required by Claim 22.

In addition to the above, the Venkatraman reference describes a system that pre-loads classes (30)(see also Abstract, lines 4-9 & 10-14) of one or several applications (32) with an embedded VM (34). Venkatraman selects a sub-set of application binary code to embed with the virtual machine to avoid the cost of loading from disk, network or any other resources in the target device (col. 2, lines 53-61). It enables a programmer to customize the class loading to use the best resource for classes according to available resources in the target devices (14). Venkatraman's apparatus does not, however, choose binary codes according to their efficiency on a specific target device -- but only to minimize the cost of loading.

Further, while Venkatraman discloses "a virtual machine that interprets" (col. 5, lines 32-33) such does NOT mean "translates into instructions", as suggested by the Examiner. Indeed, "virtual machine that interprets" in Venkatraman actually means "executes a sequence of already existing instruction to realize the referenced byte-code" (col. 5, lines 29-34). As such, there is NO generation of instructions when a virtual machine interprets.

As such, Venkatraman fails to teach or suggest, a method for **generating program source code for translating high level code into instructions for a target processor.** comprising: "**generating program source code for translating high level code into instructions for said target processor from said one or more program code modules**", as required by Claim 1, OR a method for **creating program source code for translating between high level code and instructions for a target processor,** comprising: "**forming program source code for translating high level code into instructions for said target processor from said selected one or more predefined program code modules**", as required by Claim 9, OR a data processing apparatus for **creating program source code for translating between high level code and instructions for a target processor,** the data processing apparatus being configured to: "**create program source code for translating high level code into instructions for said**

target processor from said derived one or more program code modules", as required by Claim 11.

Venkatraman further fails to teach or suggest an apparatus, comprising **at least one program source code module** of a plurality of **program source code modules** for **translating between high level code and instructions for a target processor, said at least one program source code module** corresponding to a characteristic of said target processor and being selected from said plurality of program code modules, as required by Claim 13, OR a processor, configured in accordance with **program source code** comprising at least one **program source code module** of a plurality of **program source code modules,** for **translating between high level code and instructions for a target processor,** said at least one **program source code module** being in accordance with a characteristic of said target processor and selected from said plurality of **program source code modules,** as required by Claim 22, OR a processor, configured by **program source code** comprising an **agglomeration of two or more program source code modules** of said plurality of said program code modules, as required by Claim 23, OR a system comprising a first and second processor, said first and second processor configured in accordance with program code comprising **at least two program source code modules,** wherein the **first of said at least two program source code modules is arranged to translate high level code to instructions for said first processor** and **a second of said at least two program source code modules is arranged to translate high level code to instructions** for said second processor, as required by Claim 24.

The Davis reference cited by the Examiner fails to provide any teaching or suggestion that would lead one having ordinary skill in the art to combine the teaching of Davis and Venkatraman in order to arrive at the teaching of the present invention. Even were Davis to teach or suggest a combination of the teaching of the two references, Davis fails to teach or suggest the above high-lighted elements in Claims 1, 9, 11, 13, 22, 23 or 24. As such, Davis fails to provide any teaching or suggestion that would lead one having

ordinary skill in the art to combine the teaching of Davis and Venkatraman in order to arrive at the teaching of the present invention.

In proceedings before the Patent and Trademark Office, "the Examiner bears the burden of establishing a prima facie case of obviousness based upon the prior art". *In re Fritch*, 23 USPQ2d 1780, 1783 (Fed. Cir. 1992) (citing *In re Piasecki*, 745 F.2d 1468, 1471-72, 223 USPQ 785, 787-88 (Fed. Cir. 1984). "The Examiner can satisfy this burden **only by showing some objective teaching in the prior art or that knowledge generally available to one of ordinary skill in the art would lead that individual to combine the relevant teachings of the references**", *In re Fritch*, 23 USPQ2d 1780, 1783 (Fed. Cir. 1992)(citing *In re Fine*, 837 F.2d 1071, 1074, 5 USPQ2d 1596, 1598 (Fed. Cir. 1988)(citing *In re Lalu*, 747 F.2d 703, 705, 223 USPQ 1257, 1258 (Fed. Cir. 1988)).

Although couched in terms of combining teachings found in the prior art, the same inquiry must be carried out in the context of a purported obvious "modification" of the prior art. **The mere fact that the prior art may be modified in the manner suggested by the Examiner does not make the modification obvious unless the prior art suggested the desirability of the modification.** In re Gordon, 733 F.2d at 902, 221 USPQ at 1127. Moreover, **it is impermissible to use the claimed invention as an instruction manual or "template" to piece together the teachings of the prior art so that the claimed invention is rendered obvious.** In re Gorman, 933 F.2d 982, 987, 18 USPQ2d 1885, 1888 (Fed.Cir.1991). See also Interconnect Planning Corp. v. Feil, 774 F.2d 1132, 1138, 227 USPQ 543, 547 (Fed.Cir.1985).

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable

expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest ALL the claim limitations. (MPEP § 2143). Applicants respectfully submit that the Examiner has failed to establish all three criteria.

Obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either explicitly or implicitly in the references themselves or in the knowledge generally available to one of ordinary skill in the art. "The test for an implicit showing is what the combined teachings, knowledge of one of ordinary skill in the art, and the nature of the problem to be solved as a whole would have suggested to those of ordinary skill in the art." *In re Kotzab,* 217 F.3d 1365, 1370, 55 USPQ2d 1313, 1317 (Fed. Cir. 2000). See also *In re Lee,* 277 F.3d 1338, 1342-44, 61 USPQ2d 1430, 1433-34 (Fed. Cir. 2002) (discussing the importance of relying on objective evidence and making specific factual findings with respect to the motivation to combine references); *In re Fine,* 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988); *In re Jones,* 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). Thus, Claims 1, 9, 11, 13, 22, 23 or 24 are patentable under 35 U.S.C. § 103(a) over U.S. Patent 6,571,388 B1 to Venkatraman et al., in view of U.S. Patent 6,230,307 B1 to Davis et al.

Claims 2-8 stand allowable as depending (directly or indirectly) from allowable Claim 1 and including further limitations not taught or suggested by the references of record. Claim 10 stands allowable as depending from allowable Claim 9 and including further limitations not taught or suggested by the references of record. Claim 12 stands allowable as depending from allowable Claim 11 and including further limitations not taught or suggested by the references of record. Claims 14, 15 and 18-21 stand allowable as depending (directly or indirectly) from allowable Claim 13 and including further limitations not taught or suggested by the references of record.

Claim 2 further defines the method according to claim 1, by generating program source code for **translating high level code into instructions for one of a plurality of target processors**. Claim 2 is allowable for the same reasons provided above in support of the allowability of Claim 1. Moreover, the Venkatraman and Davis references fail, alone or in combination, to teach or suggest this additional teaching in combination with the steps of Claim 1.

Claim 3 further defines the method according to claim 1, by further comprising forming agglomerated program source code from a plurality of program code modules in accordance with said program code characteristic. In addition to the reasons provided above for the allowance of Claim 1, there is no teaching whatsoever in Venkatraman that would lead one having ordinary skill in the art to "form agglomerated program source code from a plurality of program code modules in accordance with said desired program code characteristic. Similarly, there is no teaching in Davis that will overcome this deficiency in Venkatraman.

Claim 4 further defines the method according to claim 1, by further comprising deriving said program code modules in accordance with a desired functionality for said target processor. Claim 4 is allowable for the same reasons provided above in support of the allowability of Claim 1. Moreover, the Venkatraman and Davis references fail, alone or in combination, to teach or suggest this additional teaching in combination with the steps of Claim 1.

Claim 5 further defines the method according to claim 2, wherein: "said step of determining comprises determining respective program code characteristics for respective ones of a plurality of target processors", "said step of deriving comprises deriving respective program code modules in accordance with said respective program code characteristics" and "said step of generating comprises generating program source code for translating high level code into instructions for said target processors from said

program code modules". In addition to the reasons provided above for the allowance of Claim 2, Venkatraman discloses providing custom pre-loaded classes for application program for a single target device – not one of a plurality of target processors. Moreover, the Venkatraman and Davis references, alone or in combination, fail to teach or suggest multiple target processors.

Claim 6 further defines the method according to claim 1, wherein said step of deriving comprises selecting one or more pre-defined program code modules in accordance with said program code characteristic from a plurality of available program code modules. Claim 6 is allowable for the same reasons provided above in support of the allowability of Claim 1. Moreover, the Venkatraman and Davis references fail, alone or in combination, to teach or suggest this additional teaching in combination with the steps of Claim 1.

Claim 7 further defines the method according to claim 1, wherein said program code provides a virtual machine for said target processor. Claim 7 is allowable for the same reasons provided above in support of the allowability of Claim 1. Moreover, in Venkatraman, the virtual machine exists BEFORE the pre-analyzer step (see col. 5, lines 13-16 "16 are used to assemble together the virtual machine 34, a set of code for …". The tool assembles together the virtual machine and pre-load classes. As such, the Venkatraman and Davis references fail, alone or in combination, to teach or suggest the additional teaching of Claim 7 in combination with the steps of Claim 1.

Claim 8 further defines the method according to claim 1, wherein said program code comprises elements of a programming language. Claim 8 is allowable for the same reasons provided above in support of the allowability of Claim 1. Moreover, the Venkatraman and Davis references fail, alone or in combination, to teach or suggest this additional teaching in combination with the steps of Claim 1.

Claim 10 further defines the method according to claim 9, wherein said creating program source code for translating between high level code and instructions is for one of a plurality of target processors. In addition to the reasons provided above for the allowance of Claim 9, Venkatraman discloses providing custom pre-loaded classes for application program for a single target device – not one of a plurality of target processors. Moreover, the Venkatraman and Davis references, alone or in combination, fail to teach or suggest multiple target processors.

Claim 12 further defines the data processing apparatus according to claim 11, further configured for creating program source code for translating between high level code and instructions for one of a plurality of target processors. In addition to the reasons provided above for the allowance of Claim 11, Venkatraman discloses providing custom pre-loaded classes for application program for a single target device – not one of a plurality of target processors. Moreover, the Venkatraman and Davis references, alone or in combination, fail to teach or suggest multiple target processors.

Claim 14 further defines the apparatus of claim 13, by further comprising at least one additional program code modules for translating between high level code and instructions for respective ones of at least two target processors. In addition to the reasons provided above for the allowance of Claim 13, Venkatraman discloses providing custom pre-loaded classes for application program for a single target device – not one of a plurality of target processors. Moreover, the Venkatraman and Davis references, alone or in combination, fail to teach or suggest multiple target processors.

Claim 15 further defines the apparatus according to claim 14, wherein said at least two program code modules are selected from a plurality of predefined program code modules. Claim 15 is allowable for the same reasons provided above in support of the allowability of Claim 14. Moreover The Venkatraman and Davis references fail, alone or

in combination, to teach or suggest this additional teaching in combination with the steps of Claim 14.

Claim 18 further defines the apparatus according to claim 13, wherein said program source code provides a virtual machine for said target processor. Claim 18 is allowable for the same reasons provided above in support of the allowability of Claim 13. Moreover, in Venkatraman, the virtual machine exists BEFORE the pre-analyzer step (see col. 5, lines 13-16 "16 are used to assemble together the virtual machine 34, a set of code for …". The tool assembles together the virtual machine and pre-load classes. As such, the Venkatraman and Davis references fail, alone or in combination, to teach or suggest the additional teaching of Claim 13 in combination with the steps of Claim 18.

Claim 19 further defines the apparatus according to claim 14, wherein said program source code provides a virtual machine for said target processor or processors. Claim 19 is allowable for the same reasons provided above in support of the allowability of Claim 14. Moreover, in Venkatraman, the virtual machine exists BEFORE the pre-analyzer step (see col. 5, lines 13-16 "16 are used to assemble together the virtual machine 34, a set of code for …". The tool assembles together the virtual machine and pre-load classes. As such, the Venkatraman and Davis references fail, alone or in combination, to teach or suggest the additional teaching of Claim 19 in combination with the steps of Claim 14.

Claim 20 further defines the apparatus according to claim 13, wherein said program source code comprises elements of a programming language. Claim 20 is allowable for the same reasons provided above in support of the allowability of Claim 13. Moreover, the Venkatraman and Davis references fail, alone or in combination, to teach or suggest this additional teaching in combination with the steps of Claim 13.

Claim 21 further defines the apparatus according to claim 14, wherein said program source code comprises elements of a programming language. Claim 21 is allowable for the same reasons provided above in support of the allowability of Claim 14. Moreover, the Venkatraman and Davis references fail, alone or in combination, to teach or suggest this additional teaching in combination with the steps of Claim 14.

Response to Examiner's comments in Office communication:

Response to (A): The word "desired" has been deleted from Claims 1 and 3. Thus, "the program code characteristic" is the same as "a program code characteristic" recited in the previous line of Claim 1.

Response to (B): No response needed.

Response to (C): Nothing in Examiner's response provides any evidence that Venkatraman's apparatus "generates program source code **for translating high level code into instructions for a target processor**".

For the above reasons, favorable consideration of the appeal of the Final Rejection in the above-referenced application, and its reversal, are respectfully requested.

Respectfully submitted,
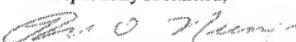
/Ronald O. Neerings/
Reg. No. 34,227
Attorney for Appellants

TEXAS INSTRUMENTS INCORPORATED
P.O. BOX 655474, M/S 3999
Dallas, Texas 75265
Phone: 972/917-5299
Fax: 972/917-4418

CLAIMS APPENDIX

1. A method for generating program source code for translating high level code into instructions for a target processor, the method comprising:

determining a program code characteristic corresponding to said target processor;

deriving one or more program code modules in accordance with said program code characteristic; and

generating program source code for translating high level code into instructions for said target processor from said one or more program code modules.

2. A method according to claim 1, for generating program source code for translating high level code into instructions for one of a plurality of target processors.

3. A method according to claim 1, comprising forming agglomerated program source code from a plurality of program code modules in accordance with said program code characteristic.

4. A method according to claim 1, further comprising deriving said program code modules in accordance with a desired functionality for said target processor.

5. A method according to claim 2, wherein:

said step of determining comprises determining respective program code characteristics for respective ones of a plurality of target processors;

said step of deriving comprises deriving respective program code modules in accordance with said respective program code characteristics; and

said step of generating comprises generating program source code for translating high level code into instructions for said target processors from said program code modules.

6. A method according to claim 1, wherein said step of deriving comprises selecting one or more pre-defined program code modules in accordance with said program code characteristic from a plurality of available program code modules.

7. A method according to claim 1, wherein said program code provides a virtual machine for said target processor.

8. A method according to claim 1, wherein said program code comprises elements of a programming language.

9. A method for creating program source code for translating between high level code and instructions for a target processor, comprising the steps of:

determining a program code characteristic corresponding to said target processor;

selecting one or more predefined program code modules in accordance with said program code characteristic; and

forming program source code for translating high level code into instructions for said target processor from said selected one or more predefined program code modules.

10. A method according to claim 9, wherein said creating program source code for translating between high level code and instructions is for one of a plurality of target processors.

11. A data processing apparatus for creating program source code for translating between high level code and instructions for a target processor, the data processing apparatus being configured to:

determine a program code characteristic corresponding to said target processor identifier input to said data processing apparatus;

derive one or more program code modules in accordance with said program code characteristic; and

create program source code for translating high level code into instructions for said target processor from said derived one or more program code modules.

12. The data processing apparatus according to claim 11, further configured for creating program source code for translating between high level code and instructions for one of a plurality of target processors.

13. An apparatus, comprising at least one program source code module of a plurality of program source code modules for translating between high level code and instructions for a target processor, said at least one program code module corresponding to a characteristic of said target processor and being selected from said plurality of program source code modules.

14. The apparatus of claim 13, further comprising at least one additional program code modules for translating between high level code and instructions for respective ones of at least two target processors.

15. The apparatus according to claim 14, wherein said at least two program code modules are selected from a plurality of predefined program code modules.

18. The apparatus according to claim 13, wherein said program source code provides a virtual machine for said target processor.

19. The apparatus according to claim 14, wherein said program source code provides a virtual machine for said target processor or processors.

20. The apparatus according to claim 13, wherein said program source code comprises elements of a programming language.

21. The apparatus according to claim 14, wherein said program source code comprises elements of a programming language.

22. A processor, configured in accordance with program code comprising at least one program source code module of a plurality of program source code modules, for translating between high level code and instructions for a target processor, said at least one program source code module being in accordance with a characteristic of said target processor and selected from said plurality of program source code modules.

23. A processor, configured by program code comprising an agglomeration of two or more program source code modules of said plurality of said program source code modules.

24. A system comprising a first and second processor, said first and second processor configured in accordance with program code comprising at least two program source code modules, wherein the first of said at least two program source code modules is arranged to translate high level code to instructions for said first processor and a second of said at least two program source code modules is arranged to translate high level code to instructions for said second processor.

EVIDENCE APPENDIX

No documents are being submitted with the Appeal Brief.